PROJECT REPORT

Credit Card Customer Segmentation Model

By:-

Shashwat Pandey

JIIT, Noida

TABLE OF CONTENTS

INTRODUCTION

1.1 Problem Statement

This case requires trainees to develop a customer segmentation to define marketing strategy. The sample dataset summarizes the usage behavior of about 9000 active credit card holders during the last 6 months. The file is at a customer level with 18 behavioral variables.

1.2 Data

Understanding of data is the very first and important step in the process of finding solution of any business problem. Here in our case our company has provided a data set with following features, we need to go through each and every variable of it to understand and for better functioning.

Size of Dataset provided:

- 8950 rows, 18 Columns

Below mentioned is a list of all the variable names with their meanings:

- CUST_ID- Credit card holder ID
- BALANCE- Monthly average balance (based on daily balance averages)
- BALANCE_FREQUENCY- Ratio of last 12 months with balance
- PURCHASES- Total purchase amount spent during last 12 months
- ONEOFF_PURCHASES -Total amount of one-off purchases
- INSTALLMENTS_PURCHASES- Total amount of installment purchases
- CASH_ADVANCE Total cash-advance amount

- PURCHASES_ FREQUENCY-Frequency of purchases (percentage of months with at least on purchase)
- ONEOFF_PURCHASES_FREQUENCY- Frequency of one-off-purchases
- PURCHASES_INSTALLMENTS_FREQUENCY- Frequency of installment purchases
- CASH_ADVANCE_ FREQUENCY- Cash-Advance frequency
- CASH_ADVANCE_TRX -Average amount per cash-advance transaction
- PURCHASES_TRX -Average amount per purchase transaction
- CREDIT_LIMIT- Credit limit
- PAYMENTS-Total payments (due amount paid by the customer to decrease their statement balance) in the period
- MINIMUM_PAYMENTS -Total minimum payments due in the period
- PRC_FULL_PAYMENT- Percentage of months with full payment of the due statement balance
- TENURE- Number of months as a customer

METHODOLOGY

2.1 Pre-Processing

When we required to build a predictive model, we require to look and manipulate the data before we start modelling which includes multiple preprocessing steps such as exploring the data, cleaning the data as well as visualizing the data through graph and plots, all these steps is combined under one shed which is Exploratory Data Analysis, which includes following steps:

- Data Exploration and Cleaning
- Missing Value Analysis
- Deriving new Key Performance Indicators(KPIs)
- Outlier Treatment
- Insights from New KPIs and Data Visualization

2.2 Applying Machine Learning Algorithms

Once all the Pre-Processing steps has been done on our data set, we will now further move to our next step which is applying ML Algorithms. ML Algorithms plays an important role to find out the good inferences from the data. Choice of models depends upon the problem statement and data set. As per our problem statement and dataset, we are identifying marketable segments using supervised learning as we are not provided with any target variable. Hence, the techniques for building non-objective segmentation model will be:

- Cluster Analysis
- KNN Techniques
- Hierarchal Clustering (Agglomerative)
- Principal Component Analysis(PCA)

2.3 Checking Performance Metrics

We have to check the validating performance with 2 metrics namely:

- Elbow Criterion
- Silhouette Score

PRE-PROCESSING

3.1 Data Exploration and Cleaning

As we start with the pre-processing techniques, we will inspect the data types to find out if there are any categorical variables that may need transforming. From the above inspection we observe that all features are numeric except for CUST_ID. But since we don't need this feature to train the model no such transformation will be done.

```
[ ] cc.describe()
```

	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUENCY	ONEOFF_PURCHASES_FREQUENCY	PUR(
count	8950.000000	8950.000000	8950.000000	8950.000000	8950.000000	8950.000000	8950.000000	8950.000000	
mean	1564.474828	0.877271	1003.204834	592.437371	411.067645	978.871112	0.490351	0.202458	
std	2081.531879	0.236904	2136.634782	1659.887917	904.338115	2097.163877	0.401371	0.298336	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	128.281915	0.888889	39.635000	0.000000	0.000000	0.000000	0.083333	0.000000	
50%	873.385231	1.000000	361.280000	38.000000	89.000000	0.000000	0.500000	0.083333	
75%	2054.140036	1.000000	1110.130000	577.405000	468.637500	1113.821139	0.916667	0.300000	
max	19043.138560	1.000000	49039.570000	40761.250000	22500.000000	47137.211760	1.000000	1.000000	

```
[ ] cc.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8950 entries, 0 to 8949
Data columns (total 18 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   CUST_ID                           8950 non-null   object
 1   BALANCE                           8950 non-null   float64
 2   BALANCE_FREQUENCY                 8950 non-null   float64
 3   PURCHASES                         8950 non-null   float64
 4   ONEOFF_PURCHASES                  8950 non-null   float64
 5   INSTALLMENTS_PURCHASES            8950 non-null   float64
 6   CASH_ADVANCE                      8950 non-null   float64
 7   PURCHASES_FREQUENCY               8950 non-null   float64
 8   ONEOFF_PURCHASES_FREQUENCY        8950 non-null   float64
 9   PURCHASES_INSTALLMENTS_FREQUENCY  8950 non-null   float64
 10  CASH_ADVANCE_FREQUENCY            8950 non-null   float64
 11  CASH_ADVANCE_TRX                  8950 non-null   int64
 12  PURCHASES_TRX                     8950 non-null   int64
 13  CREDIT_LIMIT                      8949 non-null   float64
 14  PAYMENTS                          8950 non-null   float64
 15  MINIMUM_PAYMENTS                  8637 non-null   float64
 16  PRC_FULL_PAYMENT                  8950 non-null   float64
 17  TENURE                            8950 non-null   int64
dtypes: float64(14), int64(3), object(1)
memory usage: 1.2+ MB
```

When we inspected the data, we observed some anomalies such as "CASH_ADVANCE_FREQUENCY" having some frequency values <1 which can't be possible. Hence we performed data cleaning to render this error.

```
#Data Cleaning
#As we will observe now, "CASH_ADVANCE_FREQUENCY" has some frequency values <1 which can't be possible. Hence we will perform data cleaning to render this error.
cc.loc[(cc['CASH_ADVANCE_FREQUENCY']>1)]
```

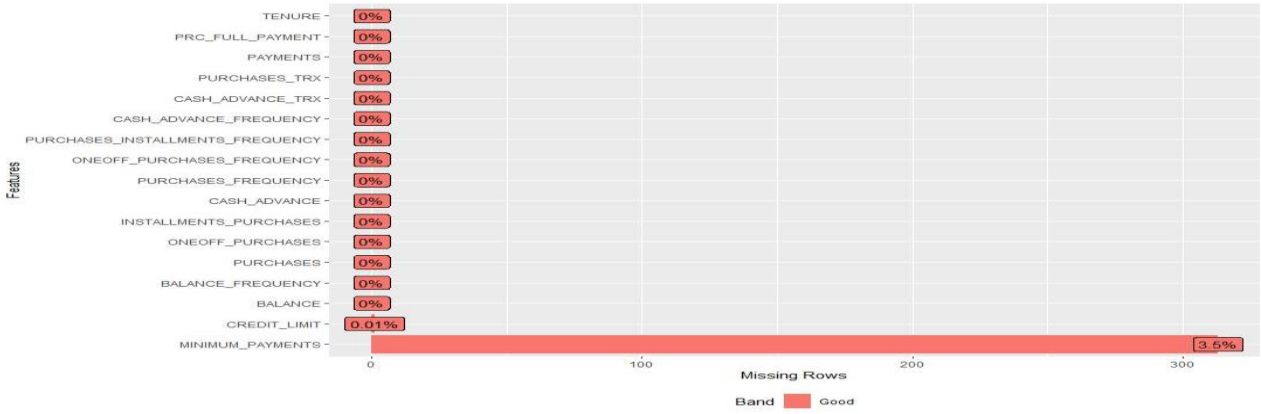
	CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUENCY	ONEOFF_PURCHASES_FREQUENCY
681	C10708	5656.069801	1.000000	362.36	362.36	0.0	7240.433194	0.250000	0.250000
1626	C11680	2876.009336	1.000000	152.61	152.61	0.0	3719.650168	0.333333	0.333333
2555	C12629	5906.184924	1.000000	141.80	141.80	0.0	1651.286918	0.125000	0.125000
2608	C12684	7801.511533	1.000000	231.40	231.40	0.0	4109.465221	0.100000	0.100000
3038	C13127	3846.742530	1.000000	0.00	0.00	0.0	1932.460679	0.000000	0.000000
3253	C13347	5709.486507	0.833333	0.00	0.00	0.0	2794.326341	0.000000	0.000000
8055	C18273	1917.895730	1.000000	285.07	285.07	0.0	6084.858872	0.363636	0.363636
8365	C18588	3857.562230	1.000000	0.00	0.00	0.0	2127.213754	0.000000	0.000000

```
[ ] #dropping the records with frequency higher that 1
    cc = cc[(cc[['CASH_ADVANCE_FREQUENCY']] <= 1).all(axis=1)]
```

3.2 Missing Value Analysis

In statistics, missing data or missing values, occur when no data value is stored for the variable in an observation. Missing data are a common occurrence and can have a significant effect on the conclusions that can be drawn from the data. Missing data can occur because of nonresponse or no information is provided for one or more items or for a whole unit. Sometimes the data is found to contain a lot of missing values. Also, missing data may reduce the precision of calculated statistics because there is less information than originally planned. Thus, missing value analysis is a very important part of data cleaning. It can be done in two ways:

- Detecting and deletion of the rows containing missing values
- Imputing the missing values by statistical methods like- mean, median or by KNN imputation or by prediction



From our analysis, we observed that the most accurate results were obtained by using mean method for prediction of missing values. Hence, we used mean method to impute the missing values.

```
65  #The closest value to the original was found with the mean method, hence we will use that.
66  cc$MINIMUM_PAYMENTS[is.na(cc$MINIMUM_PAYMENTS)] = mean(cc$MINIMUM_PAYMENTS, na.rm = T)
67  cc$CREDIT_LIMIT[is.na(cc$CREDIT_LIMIT)] = mean(cc$CREDIT_LIMIT, na.rm = T)
68
```

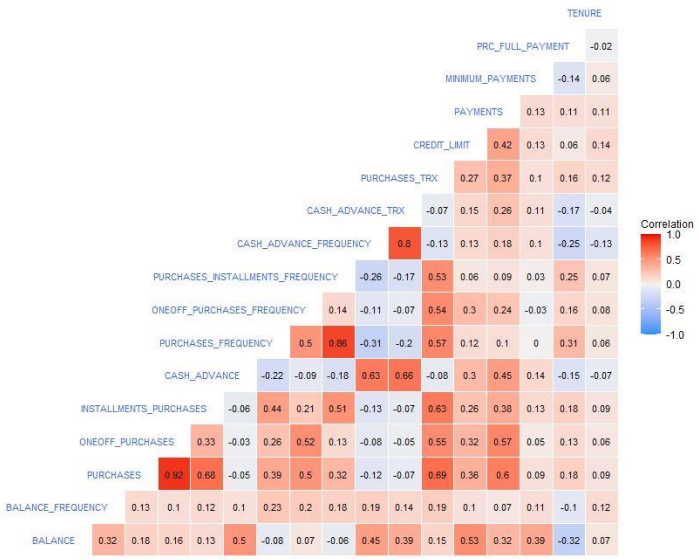
3.3 Feature Selection

In this step we would allow only to pass relevant features to further steps. We remove irrelevant features from the dataset. We do this by some statistical techniques, like we look for features which will not be helpful in predicting the target variables. Here we don't have any target variable. However after observing the dataset, we found that "CUST_ID" is a categorical variable which have no relevance in our segmentation analysis. Hence, we removed that variable.

```
[ ] #We will drop the first column "CUST_ID" as it will not help us in our custering analysis
    cc = cc.drop(['CUST_ID'], axis = 1)
    cc.head()
```

	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUENCY	ONEOFF_PURCHASES_FREQUENCY	PURCHASES_I
0	40.900749	0.818182	95.40	0.00	95.4	0.000000	0.166667	0.000000	
1	3202.467416	0.909091	0.00	0.00	0.0	6442.945483	0.000000	0.000000	
2	2495.148862	1.000000	773.17	773.17	0.0	0.000000	1.000000	1.000000	
3	1666.670542	0.636364	1499.00	1499.00	0.0	205.788017	0.083333	0.083333	
4	817.714335	1.000000	16.00	16.00	0.0	0.000000	0.083333	0.083333	

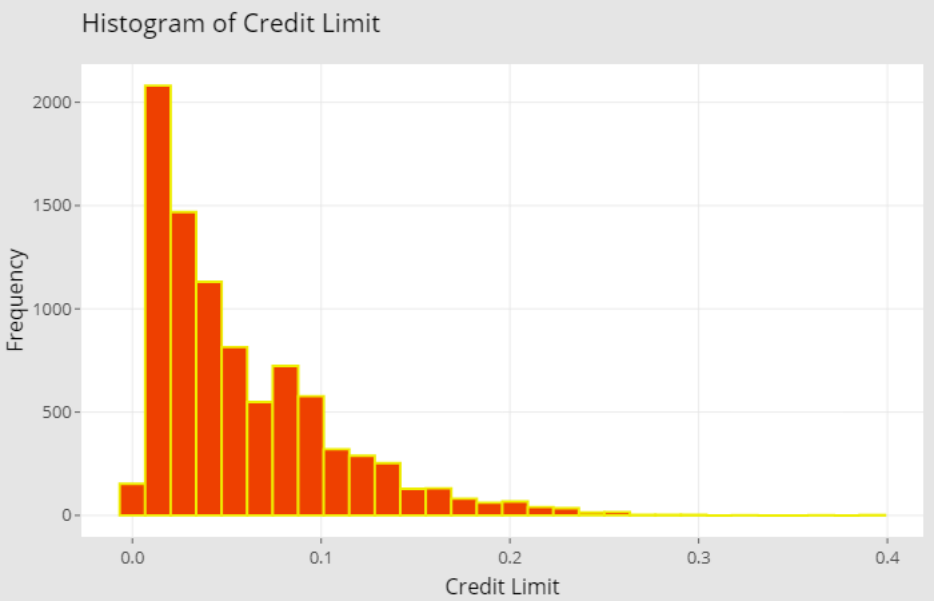
After removing the variable, we will perform correlation analysis and plot the correlation graph to see how all the variables are correlated with each other and remove if there is any “outcast” variable.

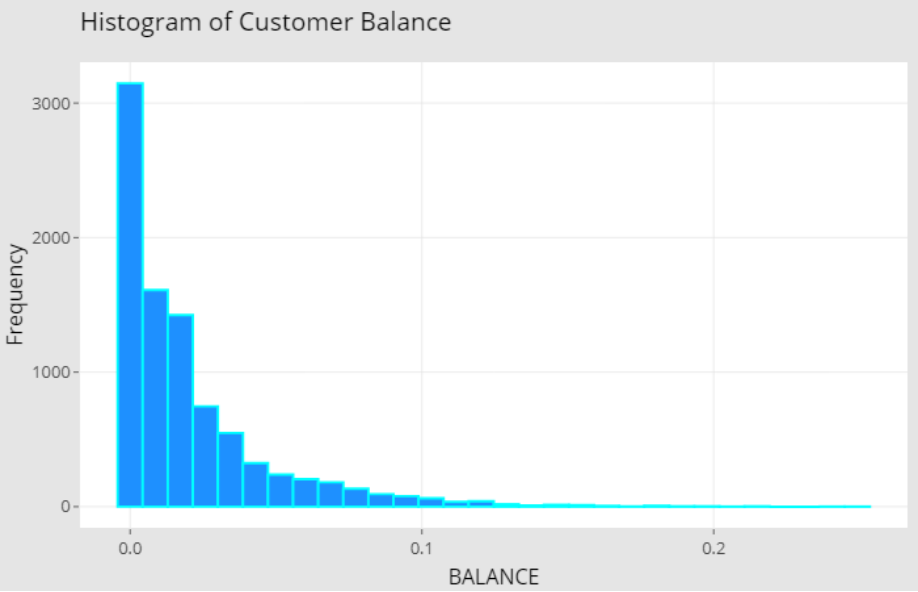


Though we find some variables not at all correlated, yet most of them are related to each other, which is why we won’t remove any variable further.

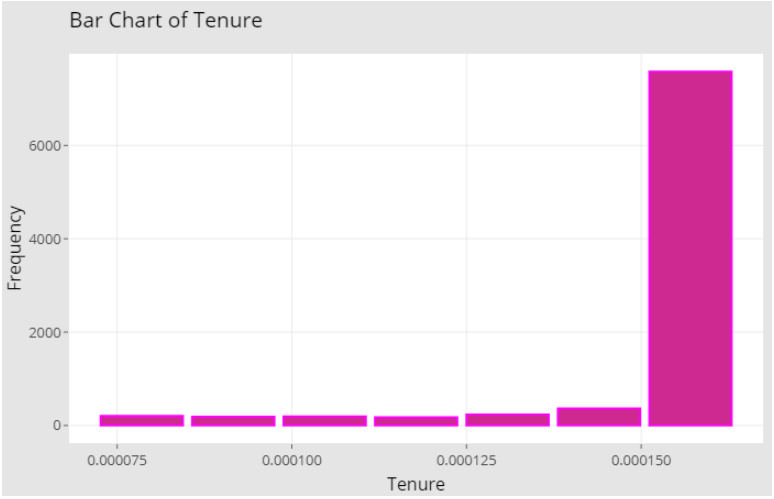
3.4 Data Visualization

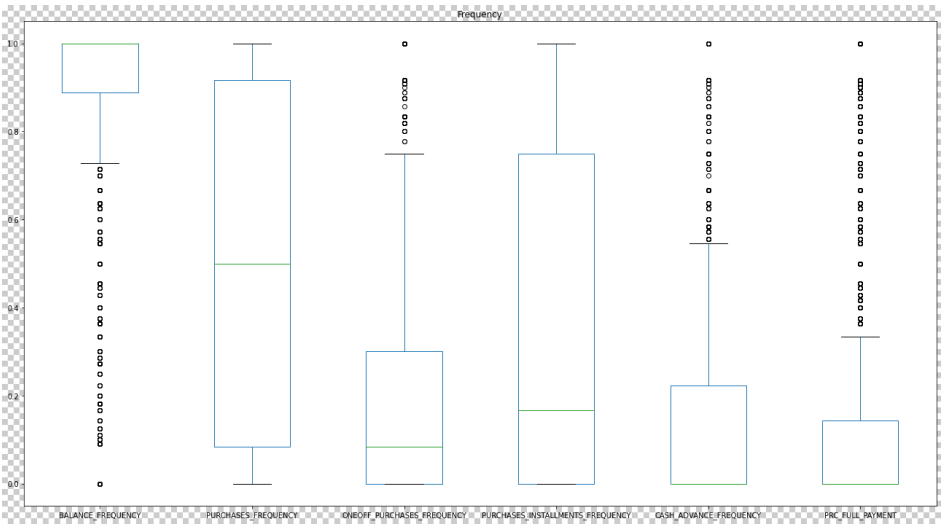
Trying to understand the dataset better with visual representation.

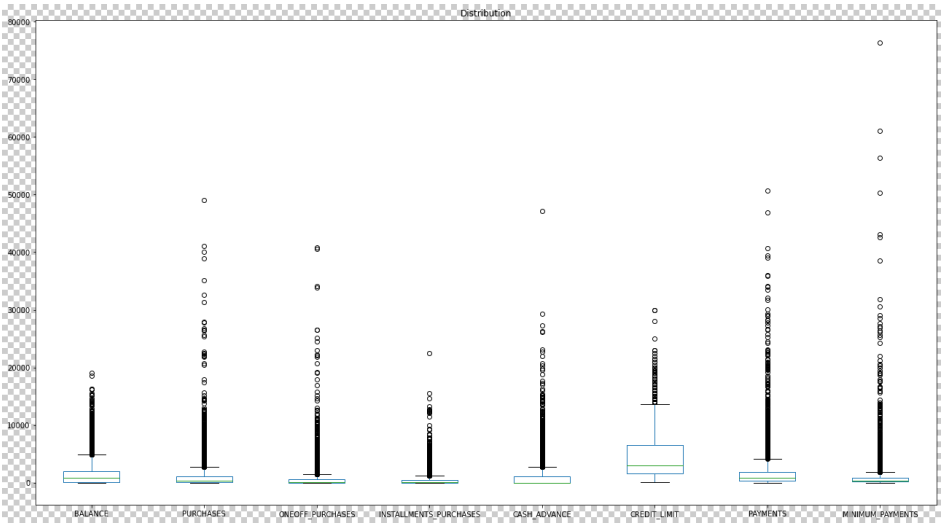


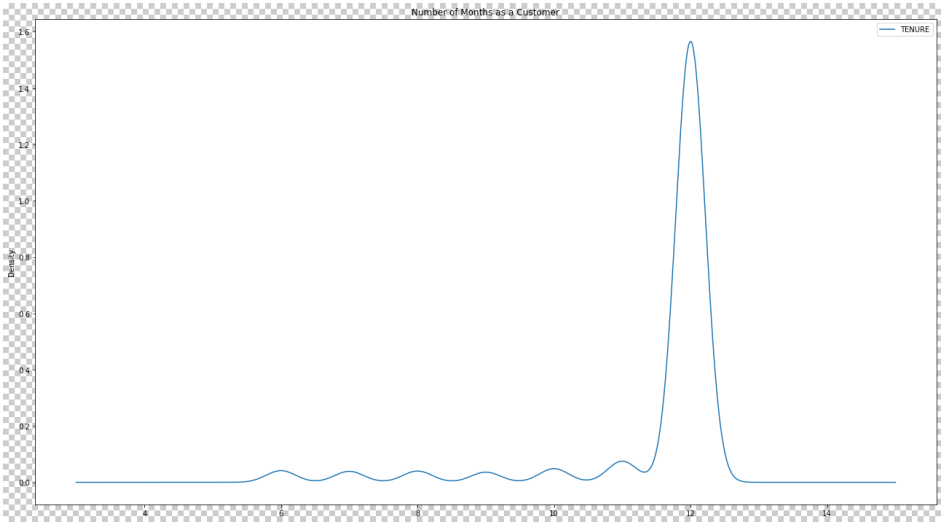

Histogram of Customer Balance
Frequency
3000
2000
1000
0
0.0
0.1
0.2
BALANCE

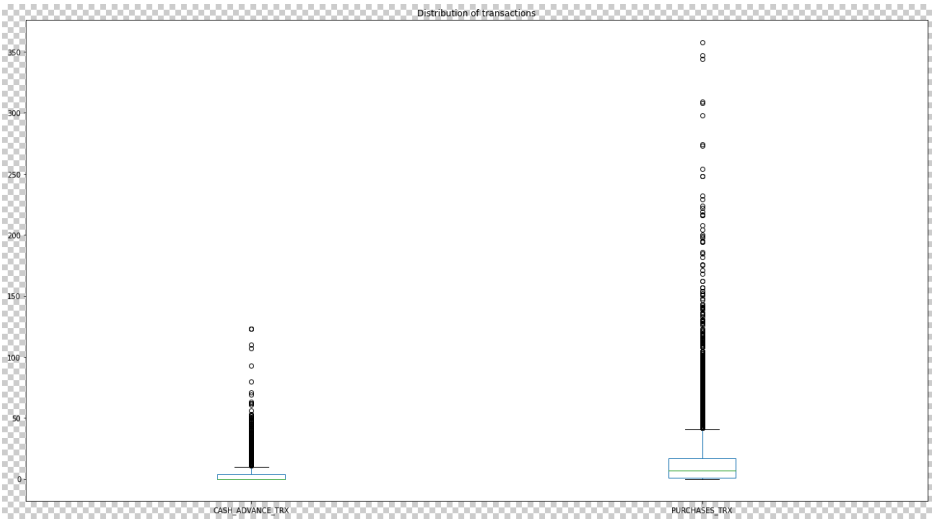

Histogram of Customer Purchase
Frequency
4000
2000
0
0.0
0.2
0.4
0.6
Purchase


Bar Chart of Tenure
Frequency
6000
4000
2000
0
0.000075
0.000100
0.000125
0.000150
Tenure


Frequency
1.0
0.8
0.6
0.4
0.2
0.0
BALANCE_FREQUENCY
PURCHASES_FREQUENCY
ONEOFF_PURCHASES_FREQUENCY
PURCHASES_INSTALLMENTS_FREQUENCY
CASH_ADVANCE_FREQUENCY
PRC_FULL_PAYMENT


Distribution
80000
70000
60000
50000
40000
30000
20000
10000
0
BALANCE
PURCHASES
ONEOFF_PURCHASES
INSTALLMENTS_PURCHASES
CASH_ADVANCE
CREDIT_LIMIT
PAYMENTS
MINIMUM_PAYMENTS


Number of Months as a Customer
TENURE
Density
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
4
6
8
10
12
14



3.5 Deriving new Key Performance Indicators (KPIs)

- Monthly_avg_purchase and Cash Advance Amount

```
[ ] #Monthly_avg_purchas
    cc['Monthly_avg_purchase']=cc['PURCHASES']/cc['TENURE']

    #Monthly_cash_advance Amount
    cc['Monthly_cash_advance']=cc['CASH_ADVANCE']/cc['TENURE']
```

```
[ ] cc['TENURE'].head()
```

```
0    12
1    12
2    12
3    12
4    12
Name: TENURE, dtype: int64
```

```
[ ] cc['Monthly_avg_purchase'].head()
```

```
0     7.950000
1     0.000000
2    64.430833
3   124.916667
4     1.333333
Name: Monthly_avg_purchase, dtype: float64
```

```
[ ] cc['PURCHASES'].head()
```

```
0     95.40
1      0.00
2    773.17
3   1499.00
4     16.00
Name: PURCHASES, dtype: float64
```

```
[ ] cc['Monthly_cash_advance'].head()
```

```
0     0.000000
1    536.912124
2     0.000000
3    17.149001
4     0.000000
Name: Monthly_cash_advance, dtype: float64
```

```
[ ] cc[cc['ONEOFF_PURCHASES']==0]['ONEOFF_PURCHASES'].count()
```

```
4299
```

- Purchase_Type
  To find what type of purchases customers are making on credit card.

```
[ ]  #Purchase Type: finding what type of purchases customers are making on credit card
     #From the data given to us we find that there are 4 types of purchase behaviours in the data set. Hence, we will derive a categorical variable based on their beh

     def purchase(cc):
         if (cc['ONEOFF_PURCHASES']==0) & (cc['INSTALLMENTS_PURCHASES']==0):
             return 'none'
         if (cc['ONEOFF_PURCHASES']>0) & (cc['INSTALLMENTS_PURCHASES']>0):
              return 'both_oneoff_installment'
         if (cc['ONEOFF_PURCHASES']>0) & (cc['INSTALLMENTS_PURCHASES']==0):
             return 'one_off'
         if (cc['ONEOFF_PURCHASES']==0) & (cc['INSTALLMENTS_PURCHASES']>0):
             return 'installment'


     cc['purchase_type']=cc.apply(purchase,axis=1)
```

```
[ ]  cc['purchase_type'].value_counts()
```

```
     both_oneoff_installment    2774
     installment                2260
     none                       2039
     one_off                    1869
     Name: purchase_type, dtype: int64
```

  What we gained from this insight was that there are 4 types of purchase behaviors observed from the customers:
  - People who only do One-Off Purchases.
  - People who only do Installments Purchases.
  - People who do both.
  - People who do none.

- Limit_Usage (balance to credit limit ratio)
  Lower value implies cutomers are maintaing thier balance properly. Lower value means good credit score.

```
[ ]  #Balance to Credit limit ratio
     cc['limit_usage']=cc.apply(lambda x: x['BALANCE']/x['CREDIT_LIMIT'], axis=1)
     cc['limit_usage'].head()
```

```
     0    0.040901
     1    0.457495
     2    0.332687
     3    0.222223
     4    0.681429
     Name: limit_usage, dtype: float64
```

- Payment to minimum payments Ratio

```
[ ] #Payment to minimum payments Ratio
    cc['payment_minpay']=cc.apply(lambda x:x['PAYMENTS']/x['MINIMUM_PAYMENTS'],axis=1)
    cc['payment_minpay'].describe()
```

```
count    8942.000000
mean        9.042468
std       118.229543
min         0.000000
25%         0.907694
50%         2.018243
75%         6.050809
max      6840.528861
Name: payment_minpay, dtype: float64
```

```
[ ] cc.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8942 entries, 0 to 8949
Data columns (total 22 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   BALANCE                           8942 non-null   float64
 1   BALANCE_FREQUENCY                 8942 non-null   float64
 2   PURCHASES                         8942 non-null   float64
 3   ONEOFF_PURCHASES                  8942 non-null   float64
 4   INSTALLMENTS_PURCHASES            8942 non-null   float64
 5   CASH_ADVANCE                      8942 non-null   float64
 6   PURCHASES_FREQUENCY               8942 non-null   float64
 7   ONEOFF_PURCHASES_FREQUENCY        8942 non-null   float64
 8   PURCHASES_INSTALLMENTS_FREQUENCY  8942 non-null   float64
 9   CASH_ADVANCE_FREQUENCY            8942 non-null   float64
 10  CASH_ADVANCE_TRX                  8942 non-null   int64
 11  PURCHASES_TRX                     8942 non-null   int64
 12  CREDIT_LIMIT                      8942 non-null   float64
 13  PAYMENTS                          8942 non-null   float64
 14  MINIMUM_PAYMENTS                  8942 non-null   float64
 15  PRC_FULL_PAYMENT                  8942 non-null   float64
 16  TENURE                            8942 non-null   int64
 17  Monthly_avg_purchase              8942 non-null   float64
 18  Monthly_cash_advance              8942 non-null   float64
 19  purchase_type                     8942 non-null   object
 20  limit_usage                       8942 non-null   float64
 21  payment_minpay                    8942 non-null   float64
dtypes: float64(18), int64(3), object(1)
memory usage: 1.6+ MB
```

3.6 Outlier Analysis

In statistics, an outlier is a data point that differs significantly from other observations. An outlier may be due to variability in the measurement or it may indicate experimental error; the latter are sometimes excluded from the data set. An outlier can cause serious problems in statistical analyses.

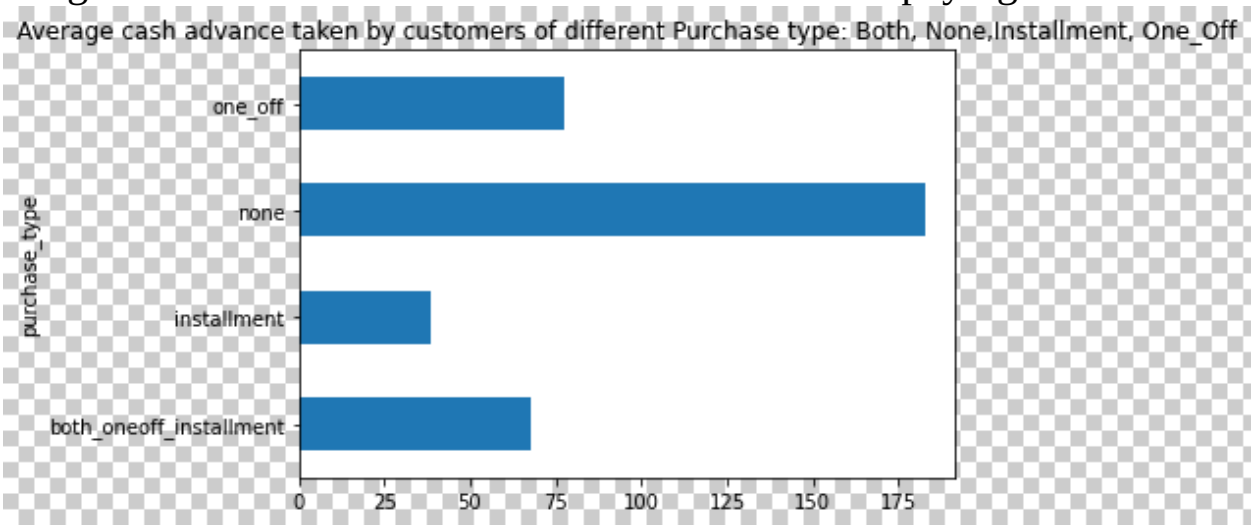
Since there are variables having extreme values, we'll be doing log transformation on the dataset to remove outlier effect. We will also drop 'purchase_type' since we won't require it for training our data.

```
[ ]  #Log tranformation
     cc_log=cc.drop(['purchase_type'],axis=1).applymap(lambda x: np.log(x+1))
     cc_log.describe()
```

	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUENCY	ONEOFF_PURCHASES_FREQUENCY	PURCHA
count	8942.000000	8942.000000	8942.000000	8942.000000	8942.000000	8942.000000	8942.000000	8942.000000	
mean	6.159660	0.619884	4.901012	3.204122	3.355403	3.314818	0.361475	0.158725	
std	2.013077	0.148642	2.916760	3.246861	3.082720	3.565004	0.277330	0.216737	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	4.860106	0.635989	3.708866	0.000000	0.000000	0.000000	0.080042	0.000000	
50%	6.771280	0.693147	5.895243	3.663562	4.505515	0.000000	0.405465	0.080042	
75%	7.624446	0.693147	7.013866	6.362183	6.152956	7.014911	0.650588	0.262364	
max	9.854515	0.693147	10.800403	10.615512	10.021315	10.760839	0.693147	0.693147	

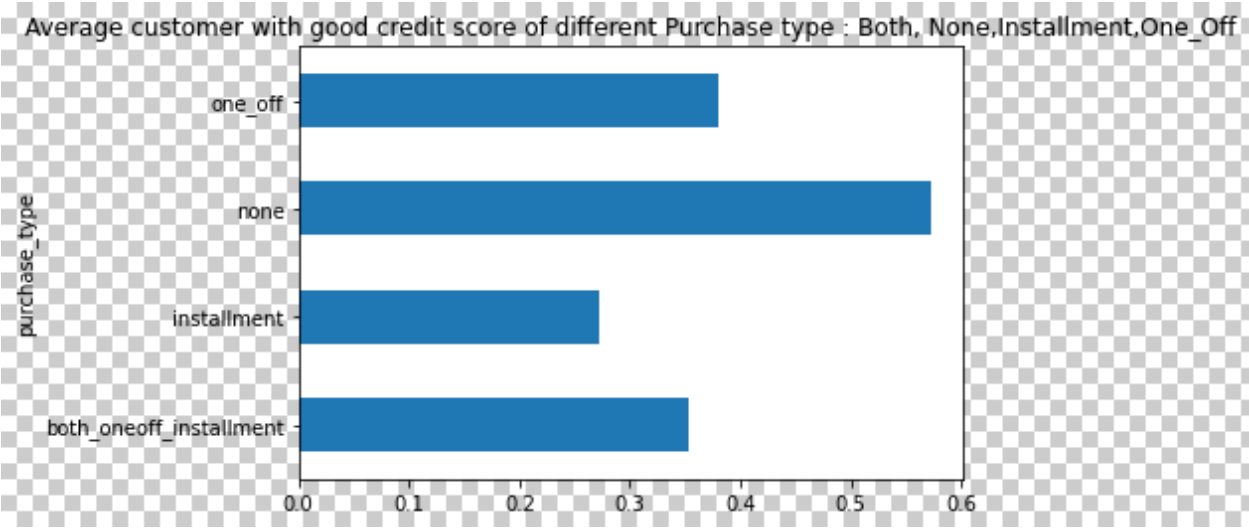
3.7 Insights from New KPIs

- Insight 1: Customers with Installment Purchases are paying dues



The graph shows us with each purchase type among 4 types as shown in the graph with the average cash advance taken by customers of different Purchase types: Both, None, Installment, One_Off. From the graph we can visualize that maximum average cash advance taken by customers is neither installment nor one_off and minimum is done by the installment customer.

- Insight 2: Customers with Installment purchases have a good credit course



From the graph we can visualize that Customers with installment purchases have good credit score. Because Lower value implies customers are maintaining their balance properly. Lower value means good credit score

3.8 Feature Scaling/Applying PCA

With the help of principal component analysis we will reduce features. PCA transforms a large set of variables into a smaller one that still contains most of the information in the large set. Reducing the number of variables of a data.

The main idea of principal component analysis (PCA) is to reduce the dimensionality of a data set consisting of many variables correlated with each other, either heavily or lightly, while retaining the variation present in the dataset, up to the maximum extent.

The dataset on which PCA technique is to be used must be scaled (we have used as cc_scaled from standardization). The results are also sensitive to the relative scaling. As a layman, it is a method of summarizing data.

Before applying PCA we will standardize data to avoid effect of scale on our result. Centering and Scaling will make all features with equal weight.

```
[ ] #Results for PCA of 5 components
    col_list=cc_dummy.columns
    pd.DataFrame(pc_final.components_.T, columns=['PC_' +str(i) for i in range(5)],index=col_list)
    #So below data gave us eigen vector for each component we had all eigen vector value very small we can remove those variable but in our case its not
```

	PC_0	PC_1	PC_2	PC_3	PC_4
BALANCE_FREQUENCY	0.030226	0.240037	-0.260611	-0.351663	-0.230124
ONEOFF_PURCHASES	0.214402	0.405219	0.240086	0.000981	-0.022654
INSTALLMENTS_PURCHASES	0.311964	-0.097820	-0.316143	0.086801	-0.002676
PURCHASES_FREQUENCY	0.345861	0.015582	-0.162695	-0.075993	0.115369
ONEOFF_PURCHASES_FREQUENCY	0.214910	0.361734	0.164005	0.036121	-0.050206
PURCHASES_INSTALLMENTS_FREQUENCY	0.295361	-0.111560	-0.330375	0.021799	0.024847
CASH_ADVANCE_FREQUENCY	-0.214338	0.286993	-0.279895	0.092874	0.358916
CASH_ADVANCE_TRX	-0.229181	0.292465	-0.285496	0.098804	0.332939
PURCHASES_TRX	0.355591	0.106350	-0.102440	-0.055665	0.104782
Monthly_avg_purchase	0.346172	0.140846	0.024291	-0.080377	0.193911
Monthly_cash_advance	-0.243703	0.265308	-0.257753	0.131205	0.268618
limit_usage	-0.145896	0.235515	-0.249071	-0.436408	-0.185848
payment_minpay	0.119117	0.025894	0.131239	0.591671	0.216868
both_oneoff_installment	0.241397	0.274464	-0.131587	0.253642	-0.340365
installment	0.081934	-0.443567	-0.209541	-0.190830	0.352715
none	-0.310500	-0.003581	-0.097235	0.245788	-0.341309
one_off	-0.041810	0.165533	0.473996	-0.338219	0.362423

```
#Factor Analysis : variance explained by each component-
pd.Series(pc_final.explained_variance_ratio_,index=['PC_'+ str(i) for i in range(5)])
```

```
PC_0    0.401947
PC_1    0.180556
PC_2    0.147469
PC_3    0.081345
PC_4    0.065321
dtype: float64
```

APPLYING MACHINE LEARNING ALGORITHMS

Segmentation in marketing is a technique used to divide customers or other entities into groups based on attributes such as behavior or demographics. It is useful to identify segments of customers who may respond in a similar way to specific marketing techniques such as email subject lines or display advertisements. As it gives businesses the ability to tailor marketing messages and timing to generate better response rates and provide improved consumer experiences.
There are two broad set of methodologies for segmentation: Objective (supervised) and Non- Objective (unsupervised) segmentation methodologies. As the name indicates, a supervised methodology requires the objective to be stated as the basis for segmentation.
But in our current project after understanding the dataset, the segments are different with respect to the "generic profile" of observations belonging to each segment, but not with regards to any specific outcome of interest (i.e. no target label is available). Hence, we will be going with the unsupervised machine learning techniques.
By analyzing the dataset with unsupervised machine learning algorithm, we have to determine based on the dataset available what are the marketing strategy is there.

4.1 K- Means Clustering

K-means clustering is one of the simplest and popular unsupervised machine learning algorithms. It is an iterative algorithm that tries to partition the dataset into pre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to only one group.
Typically, unsupervised algorithms make inferences from datasets using only input vectors without referring to known, or labelled, outcomes.
The goal of this algorithm is to find K groups in the data. The algorithm works iteratively to assign each data point to one of K groups based on the features that are provided. Data points are clustered based on feature similarity. The results of the K-means clustering algorithm are:

- The centroids of the K clusters, which can be used to label new data
- Labels for the training data (each data point is assigned to a single cluster)

K-means works by defining spherical clusters that are separable in a way so that the mean value converges towards the cluster center. Because of this, K-Means may underperform sometimes.
To process the learning data, the K-means algorithm in data mining starts with a first group of randomly selected centroids, which are used as the beginning points for every cluster, and then performs iterative (repetitive) calculations to optimize the positions of the centroids.
It halts creating and optimizing clusters when either:

- The centroids have stabilized — there is no change in their values because the clustering has been successful.
- The defined number of iterations has been achieved.

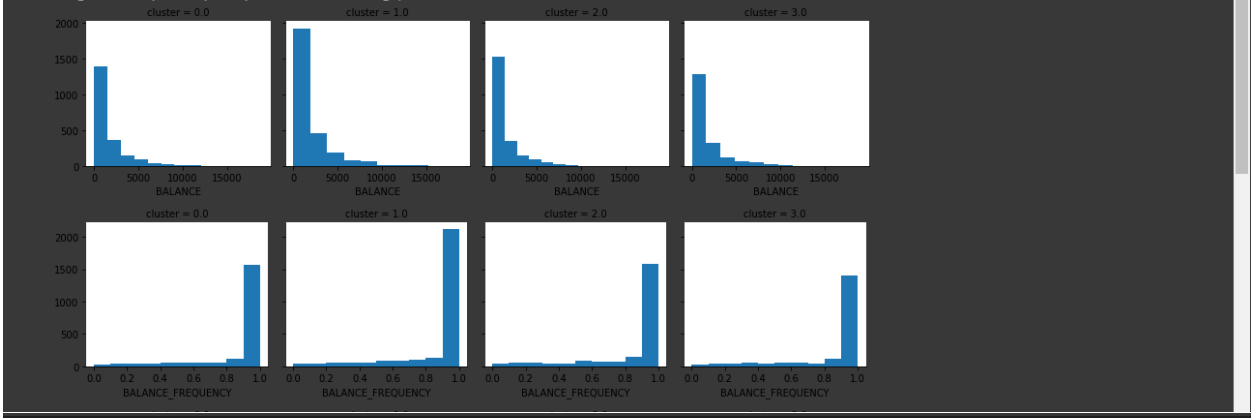
4.2 Clustering

Clustering is one of the most common exploratory data analysis technique used to get an intuition about the structure of the data.
It can be defined as the task of identifying subgroups in the data such that data points in the same subgroup (cluster) are very similar while data points in different clusters are very different. In other words, we try to find homogeneous subgroups within the data such that data points in each cluster are as similar as possible according to a similarity measure such as Euclidean-based distance or correlation-based distance. The decision of which similarity measure to use is application-specific.
Clustering analysis can be done on the basis of features where we try to find subgroups of samples based on features or on the basis of samples where we try to find subgroups of features based on samples.
We'll cover here clustering based on features. Clustering is used in market segmentation; where we try to find customers that are similar to each other whether in terms of behaviors or attributes, image

segmentation/compression; where we try to group similar regions together, document clustering based on topics, etc.
Unlike supervised learning, clustering is considered an unsupervised learning method since we don't have the ground truth to compare the output of the clustering algorithm to the true labels to evaluate its performance. We only want to try to investigate the structure of the data by grouping the data points into distinct subgroups.
Cluster analysis is an exploratory analysis that tries to identify structures within the data. Cluster analysis is also called segmentation analysis or taxonomy analysis. More specifically, it tries to identify homogenous groups of cases if the grouping is not previously known.

```
[ ] #Interpreting clusters
    for c in clusters:
        grid= sns.FacetGrid(clusters, col='cluster')
        grid.map(plt.hist, c)
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/axisgrid.py:318: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot inter
  fig, axes = plt.subplots(nrow, ncol, **kwargs)
```

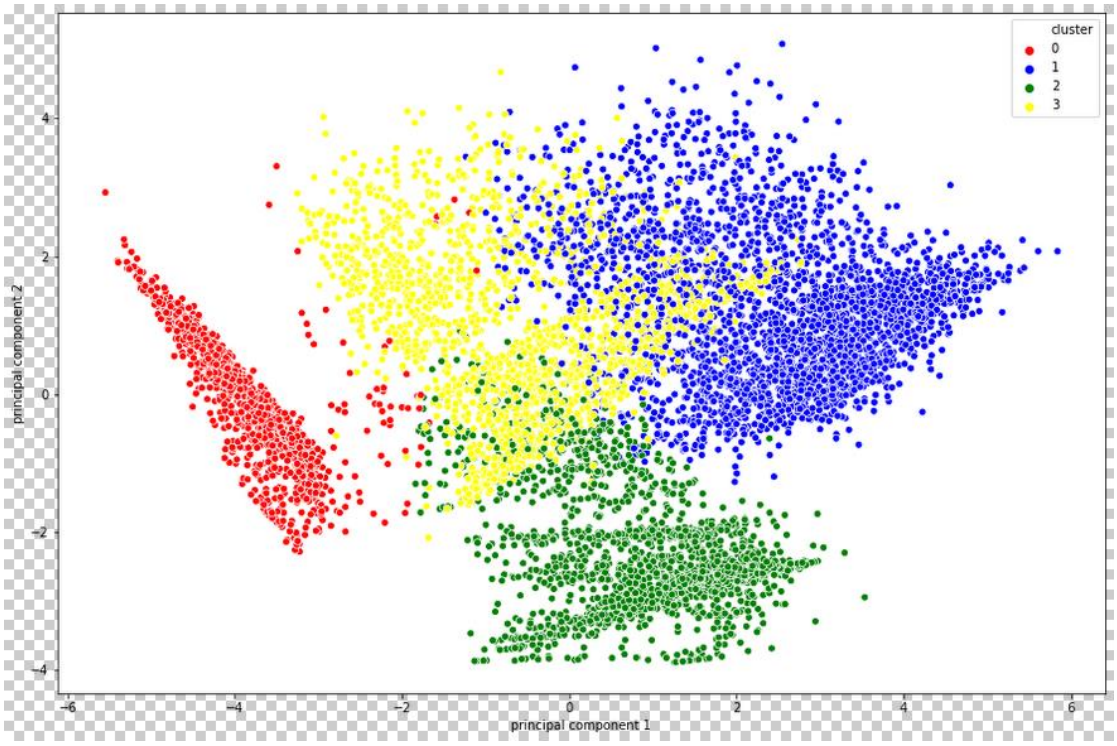


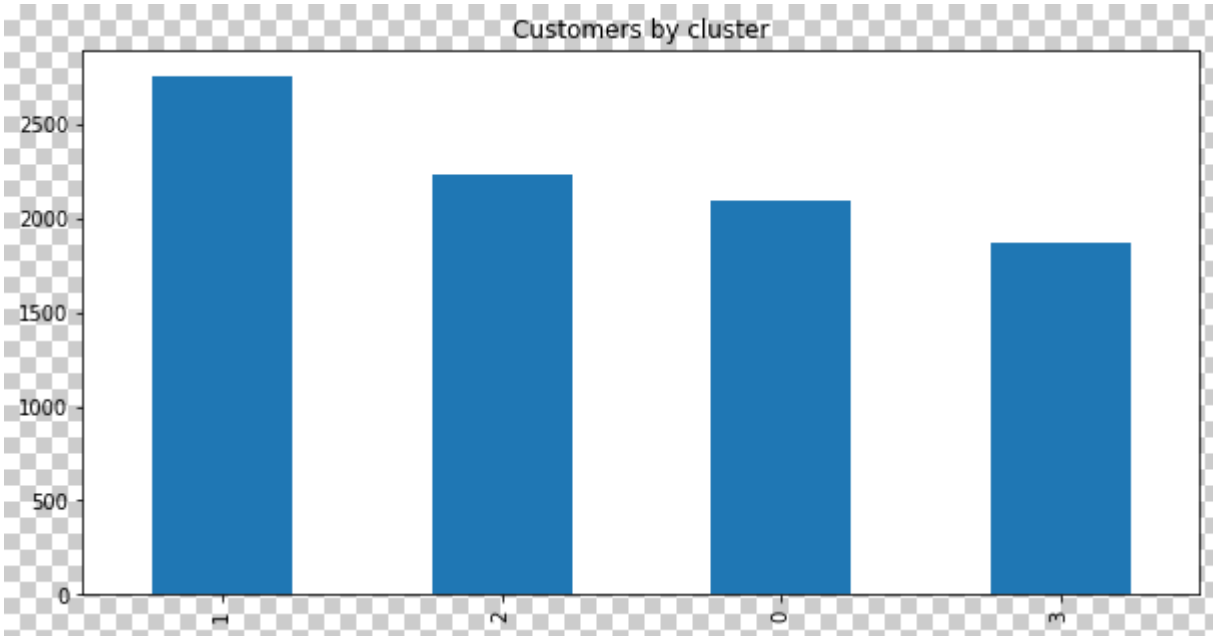

```
[ ] #Analysing Clusters
    cluster_pca = clusters.groupby('cluster').mean()
    cluster_pca
```

cluster	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUENCY	ONEOFF_PURCHASES_FREQUENCY	PUR
0.0	1558.009898	0.880551	810.042010	456.277564	354.262523	998.506370	0.466288	0.178368	
1.0	1745.439028	0.888489	1300.535252	783.646958	517.281978	1051.662181	0.518531	0.237588	
2.0	1313.817517	0.858463	828.055645	454.723757	373.518854	823.770335	0.492176	0.168370	
3.0	1596.255536	0.879218	995.290701	629.731354	365.638288	1026.735729	0.474377	0.218611	

Just looking at 'PURCHASES_FREQUENCY' we can see that the model has identified some high-frequency purchase segments, clusters 1 and 2. Let's understand the differences between these two segments to further determine why they are in separate clusters. We can see that cluster 2 has a higher number of total purchases, a higher credit limit, they make frequent one-off purchases and are more likely to pay in full.

We can draw the conclusion that these are high-value customers and therefore there will almost certainly be a difference between how you may market to these customers.
As a first iteration of the model, this appears to be identifying some useful segments. There are many ways in which we could tune the model including alternative data cleaning methods, feature engineering, dropping features with high correlation, which we have already implemented in this project.
Followed by forming clusters, we apply PCA to transform data to 2 dimensions for visualization. We won't be able to visualize the data in 17 dimensions so reducing the dimensions with PCA.
PCA transforms a large set of variables into a smaller one that still contains most of the information in the large set. Reducing the number of variables of a data.





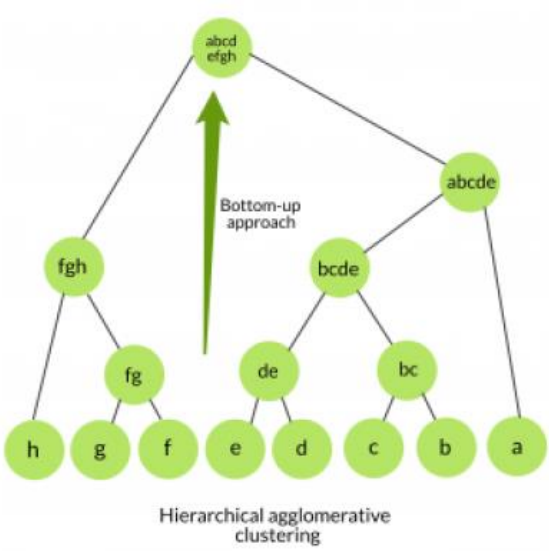
4.3 Hierarchal Clustering

Hierarchical clustering is one of the popular and easy to understand clustering technique. In data mining and statistics, hierarchical clustering analysis is a method of cluster analysis which seeks to build a hierarchy of clusters i.e. tree type structure based on the hierarchy. This clustering technique is divided into two types:

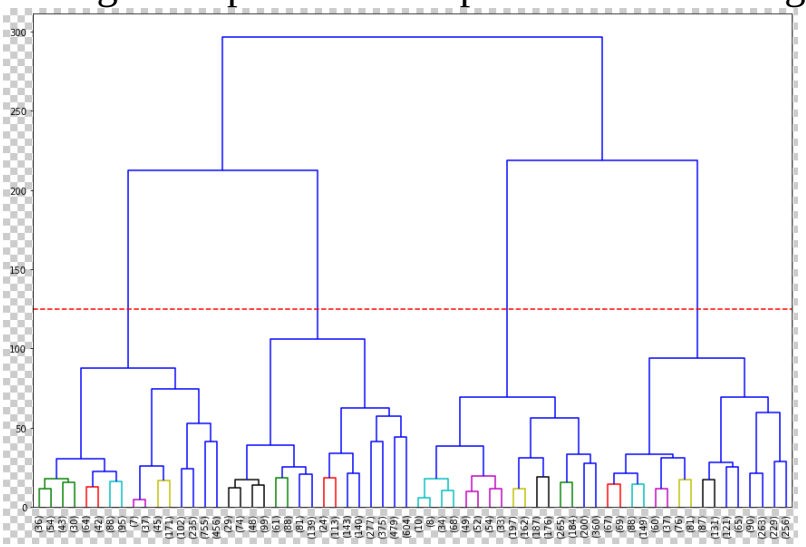
- Agglomerative
- Divisive

4.3.1 Agglomerative Clustering

Also known as bottom-up approach or hierarchical agglomerative clustering (HAC). A structure that is more informative than the unstructured set of clusters returned by flat clustering. This clustering algorithm does not require us to pre-specify the number of clusters. Bottom-up algorithms treat each data as a singleton cluster at the outset and then successively agglomerates pairs of clusters until all clusters have been merged into a single cluster that contains all data. In this technique, initially each data point is considered as an individual cluster. At each iteration, the similar clusters merge with other clusters until one cluster or K clusters are formed.



The Hierarchical clustering Technique can be visualized using a Dendrogram. A Dendrogram is a tree-like diagram that records the sequences of merges or splits which represent the Dendrogram.



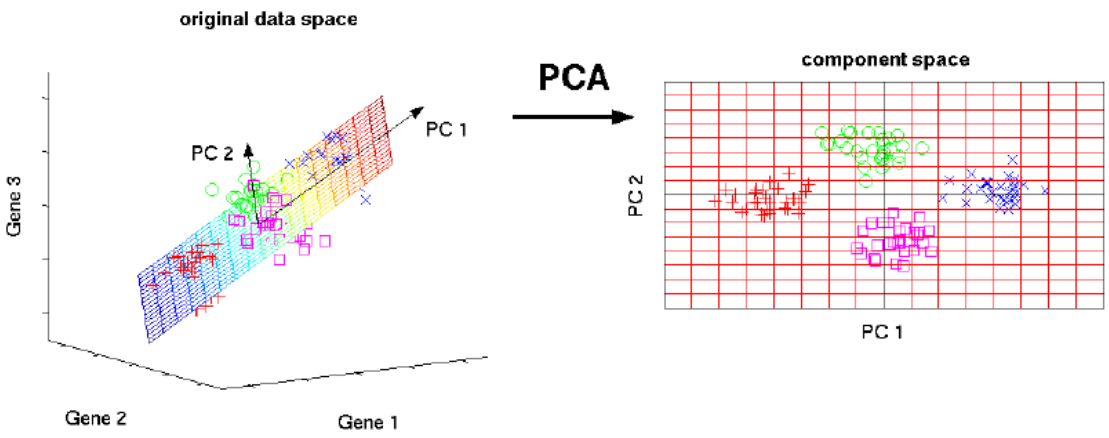
4.4 Principal Component Analysis

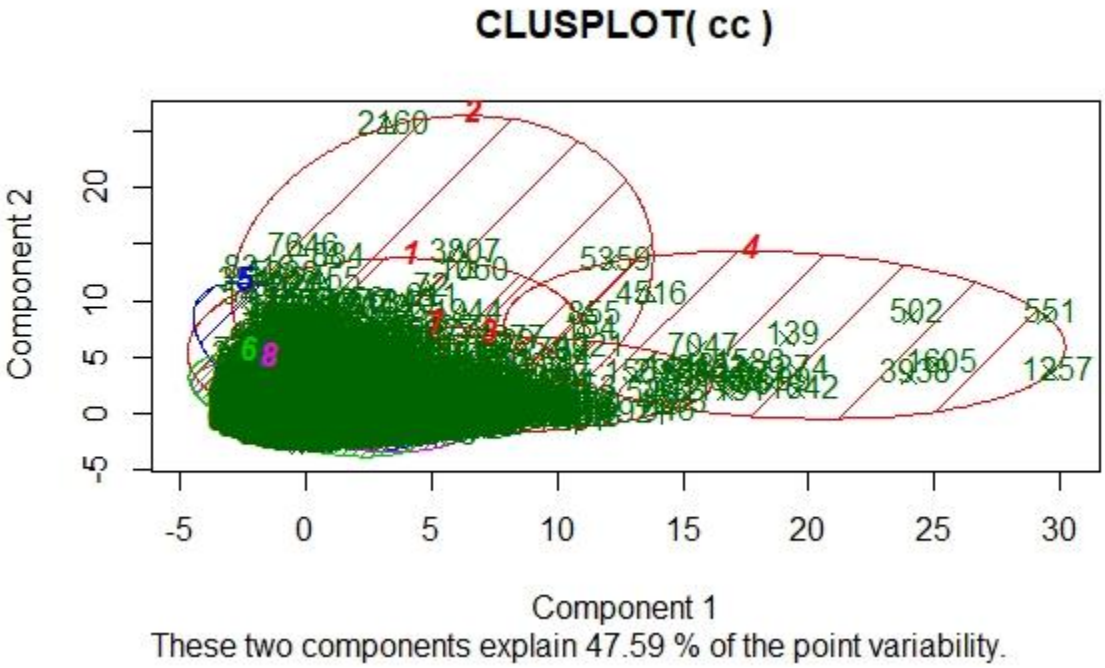
Principal Component Analysis (PCA) is a dimension-reduction tool that can be used to reduce a large set of variables to a small set that still contains most of the information in the large set.
Objectives of principal component analysis:

- Principal component analysis (PCA) is a mathematical procedure that transforms a number of (possibly) correlated variables into a (smaller) number of uncorrelated variables called principal components.
- The first principal component accounts for as much of the variability in the data as possible, and each succeeding component accounts for as much of the remaining variability as possible.
- PCA reduces attribute space from a larger number of variables to a smaller number of factors and as such is a "non-dependent" procedure (that is, it does not assume a dependent variable is specified).
- PCA is a dimensionality reduction or data compression method. The goal is dimension reduction and there is no guarantee that the dimensions are interpretable (a fact often not appreciated by (amateur) statisticians).
- To select a subset of variables from a larger set, based on which original variables have the highest correlations with the principal component.

We will be using Principal component analysis (PCA) because of the three reasons:

- We want to reduce the number of variables, but aren’t able to identify variables to completely
- Remove from consideration.
- Want to ensure variables are independent of one another.
- Comfortable making independent variables less interpretable.





PCA Analysis done on clusters in R

CHECKING PERFORMANCE METRICS

5.1 Model Evaluation

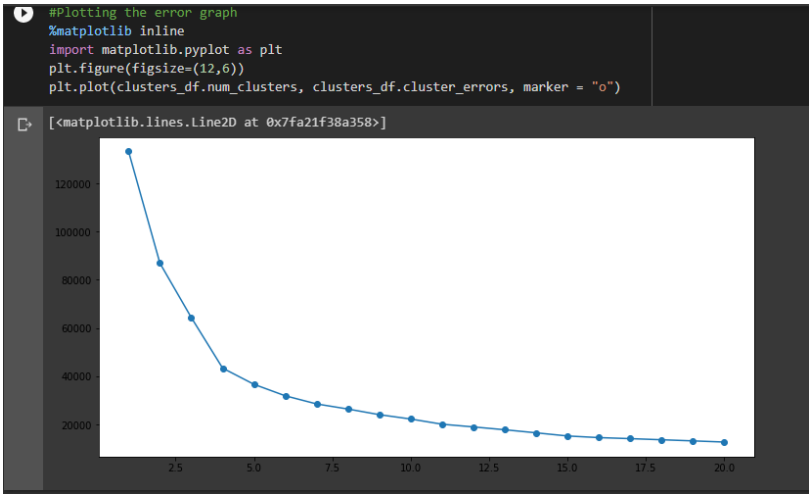
Now that we have a few models for performing the segmentation, we need to decide which one to choose and optimize it further for deployment.

5.2 Choosing K

If the true label is not known in advance, then K-Means clustering can be evaluated using Elbow Criterion, Silhouette Coefficient , cross-validation, information criteria, the information theoretic jump method, and the G-means algorithm.

5.3 Elbow Criterion

The idea behind elbow method is to run k-means clustering on a given dataset for a range of values of k (e.g k=1 to 10), for each value of k, calculate sum of squared errors (SSE). Calculate the mean distance between data points and their cluster centroid. Increasing the number of clusters (K) will always reduce the distance to data points, thus decrease this metric, to the extreme of reaching zero when K is as same as the number of data points. So the goal is to choose a small value of k that still has a low SSE.
We run the algorithm for different values of K(say K = 10 to 1) and plot the K values against SSE(Sum of Squared Errors). And select the value of K for the elbow point.



From above graph we find the elbow range. We can see that after almost 4, 5, 6 clusters adding more gives minimal benefit to the model. We are therefore going to use 5 clusters to train our model.

5.4 Silhouette Coefficient

A higher Silhouette Coefficient score relates to a model with better-defined clusters. The Silhouette Coefficient is defined for each sample and is composed of two scores:

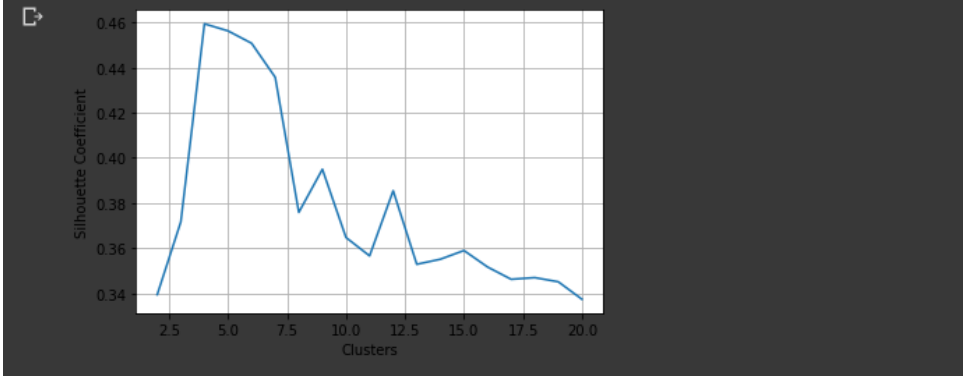
- The mean distance between a sample and all other points in the same class.
- The mean distance between a sample and all other points in the next nearest cluster.

The Silhouette Coefficient is for a single sample is then given as:

$$s = (b-a)/\max(a,b)$$

To find the optimal value of k for K Means, loop through 1...n for n_clusters in K Means and calculate Silhouette Coefficient for each sample. A higher Silhouette Coefficient indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters.

```
[ ]  #Plot the Result
     plt.plot(k_range, scores)
     plt.xlabel('Clusters')
     plt.ylabel('Silhouette Coefficient')
     plt.grid(True)
```



From metrics.silhouette_score method above graph is suggesting to choose the K value is 4 i.e. number of cluster = 4

5.5 Suggested Marketing Strategy

We are going to neglect the K-means models with 5 and 6 clusters and going to accept the model with 4 clusters. Reason being with 5 and 6 clusters behaving the similar kind with other cluster too.
Insights with 4 Clusters:

- Cluster 2 is the group of customers who have highest Monthly_avg purchases and doing both installment as well as one_off purchases, have comparatively good credit score. This group is about 31% of the total customer base
- Cluster 1 is taking maximum advance_cash and is paying comparatively less minimum payment and poor credit_score & doing no purchase transaction. This group is about 23% of the total customer base
- Cluster 0 customers are doing maximum One_Off transactions and least payment ratio and credit_score on lower side. This group is about 21% of the total customer base
- Cluster 3 customers have maximum credit score and are paying dues and are doing maximum installment purchases. This group is about 25% of the total customer base

Marketing Strategy Suggested:

- Group 2: They are potential target customers who are paying dues and doing purchases and maintaining good credit score) -- we can increase credit limit or can lower down interest rate Can be given premium card /loyality cards to increase transactions
- Group 1: They have poor credit score and taking only cash on advance. We can target them by providing less interest rate on purchase transaction
- Group 0: This group is has minimum paying ratio and using card for just one off transactions (may be for utility bills only). This group seems to be risky group.
- Group 3: This group is performing best among all as cutomers are maintaining good credit score and paying dues on time. -- Giving rewards point will make them perform more purchases.

5.6 Conclusion

By analyzing and visualizing the both concept with the Principal Component Analyzing (PCA) i.e. K Means and Agglomerative / Hierarchical Clustering both algorithm had given 4 cluster.
Almost the behavior of Agglomerative / Hierarchical Clustering is same as K Means with similar results. But only difference is there Cluster 0 and 2 are interchanged and, cluster 1 and 3 are interchanged.

So Final conclusion is that we have **accepted K Means algorithm over Agglomerative / Hierarchical Clustering**. Choosing the n_components for K with the both concept Elbow Criterion Method and Silhouette Coefficient Method which give the value 4.
We have tried to explain every concept which we have used in this project with acceptance as well as rejected. We have also explained the market strategy and also the behavior of all 4 clusters.

REFERENCES:

- https://en.wikipedia.org/
- https://scikit-learn.org/stable/modules/clustering.html#clustering (https://scikitlearn.org/stable/modules/clustering.html#clustering)
- https://learning.edwisor.com/
- https://medium.com/
- https://www.statisticshowto.datasciencecentral.com/
- https://www.theanalysisfactor.com/
- James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2013. An Introduction to Statistical Learning. Vol. 6. Springer.
- https://www.analyticsvidhya.com/blog/2016/03/practical-guide-principal-componentanalysis-python/?# (https://www.analyticsvidhya.com/blog/2016/03/practical-guideprincipal-component-analysis-python/)